IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	§	Filed: October 9, 2003
Dettinger et al.	§	
	§	Group Art Unit: 2166
Serial No.: 10/682,133	§	
	§	Examiner: Khanh B. Pham
Confirmation No.: 1361	§	

For: MODELING AND IMPLEMENTING COMPLEX DATA ACCESS OPERATIONS BASED ON LOWER LEVEL TRADITIONAL OPERATIONS

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Khanh B. Pham, or electronically transmitted via EFS-Web, on the date shown below:

February 12, 2008 /Jon K. Stewart/
Date Jon K. Stewart

APPEAL BRIEF

Dear Sir:

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2166 dated September 12, 2007, finally rejecting claims 1-17, 26-41, 50 and 51. The final rejection of claims 1-17, 26-41, 50 and 51 is appealed. This Appeal Brief is believed to be timely since it is transmitted by the due date of February 12, 2008, as set by the filing of a Notice of Appeal on December 12, 2007.

Please charge the fee of $510.00 for filing this brief to:

Deposit Account No. 09-0465 / ROC920030237US1.

TABLE OF CONTENTS

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-17, 26-41, 50 and 51 are pending in the application. Claims 1-54 were originally presented in the application. Claims 18-25, 42-49 and 52-54 have been canceled without prejudice. Claims 1-17, 26-41, 50 and 51 stand finally rejected as discussed below. The final rejections of claims 1-17, 26-41, 50 and 51 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner. No amendments to the claims were proposed after the final rejection.

Summary of Claimed Subject Matter

Claimed embodiments include methods (see claims 1-17), computer programs stored on computer readable storage media (see claims 26-41), and data processing systems (see claims 50-51) directed to managing execution of query operations in a data processing system. In one embodiment, a complex data access operation is implemented as a composite query operation, which appears as a single query operation to a requesting entity. The composite query operation is effectively composed of a sequence of single query operations implemented in an order described by one or more encapsulated implementation schemas. Rather than having the requesting entity issue a sequence of fine-grained single query operations connected by appropriate selection logic, the requesting entity can issue one composite query operation where the appropriate selection logic is already encapsulated within. *See Application*, page 10, line 23 – page 11, line 10.

A. CLAIM 1 - INDEPENDENT

Claim 1 recites a method of managing execution of query operations in a data processing system. See *Application*, page 4, lines 1-7. As claimed, the method includes issuing, by a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system, and executing the initial query operation. See *Application*, page 16, lines 10-27; FIG. 2. The method also includes determining an operation status of the initial query operation. See *Application*, page 18, line 3 – page 19, line 5; FIG. 3-4. The method also includes selecting one of the plurality of subsequent query operations based on the operation status, and performing the selected subsequent query operation. See *Application*, page 23, line 23 – page 24, line 17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The method also includes updating the operation status based on a result of the subsequent query operation, and managing execution of any remaining subsequent query operations on the basis of the updated operation status. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B.

The method also includes, upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity. See *Application*, page 17, lines 1-7.

B. CLAIM 9 - INDEPENDENT

Claim 9 recites a method of managing execution of query operations in a data processing system. See *Application*, page 4, lines 8-17. As claimed, the method includes issuing, by a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system. See *Application*, page 16, lines 10-27; FIG. 2. The method also includes providing selection logic defining a next query operation of the composite query operation to be executed. See *Application*, page 18, lines 3–15; page 23, line 23 – page 24, line 17; FIG. 3-4. The method also includes providing a plurality of failure conditions for determining when a failure of the composite query operation occurs. See *Application*, page 18, line 16 – page 20, line 4; page 24, lines 9-17; FIG. 3-4. The method also includes managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The method also includes, upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity. See *Application*, page 17, lines 1-7.

C. CLAIM 26 - INDEPENDENT

Claim 26 is directed to a computer readable storage medium containing a program which, when executed, performs a process of managing execution of query operations in a data processing system. See *Application*, page 5, lines 20-27. As claimed, the program includes receiving, from a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data

processing system, and executing the initial query operation. See *Application*, page 16, lines 10-27; FIG. 2. The program also includes determining an operation status of the initial query operation. See *Application*, page 18, line 3 – page 19, line 5; FIG. 3-4. The program also includes selecting one of the plurality of subsequent query operations based on the operation status, and performing the selected subsequent query operation. See *Application*, page 23, line 23 – page 24, line 17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The program also includes updating the operation status based on a result of the subsequent query operation, and managing execution of any remaining subsequent query operations on the basis of the updated operation status. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The program also includes, upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity. See *Application*, page 17, lines 1-7.

D. CLAIM 33 - INDEPENDENT

Claim 33 is directed to a computer readable storage medium containing a program which, when executed, performs a process of managing execution of query operations in a data processing system. See *Application*, page 5, line 28 – page 6, line 9. As claimed, the program includes receiving, from a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system. See *Application*, page 16, lines 10-27; FIG. 2. The program also includes retrieving selection logic defining a next query operation of the composite query operation to be executed. See *Application*, page 18, lines 3–15; page 23, line 23 – page 24, line 17; FIG. 3-4. The program also includes retrieving a plurality of failure conditions for determining when a failure of the composite query operation occurs. See *Application*, page 18, line 16 – page 20, line 4; page 24, lines 9-17; FIG. 3-4. The program also includes managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The

program also includes, upon determining the composite query operation has completed, storing and returning a result of the composite query operation to the requesting entity. See *Application*, page 17, lines 1-7.

E. CLAIM 50 - INDEPENDENT

Claim 50 is directed to a data processing system that includes a data repository and a composite query operations manager residing in memory for managing execution of query operations in the data processing system. See *Application*, page 7, lines 13-21. As claimed, the composite query operations manager is configured for receiving a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against the data repository, and executing the initial query operation. See *Application*, page 16, lines 10-27; FIG. 2. The composite query operations manager is also configured for determining an operation status of the initial query operation. See *Application*, page 18, line 3 – page 19, line 5; FIG. 3-4. The composite query operations manager is also configured for selecting one of the plurality of subsequent query operations based on the operation status, and performing the selected subsequent query operation. See *Application*, page 23, line 23 – page 24, line 17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The composite query operations manager is also configured for updating the operation status based on a result of the subsequent query operation, and managing execution of any remaining subsequent query operations on the basis of the updated operation status. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The composite query operations manager is also configured for, upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity. See *Application*, page 17, lines 1-7.

F. CLAIM 51 - INDEPENDENT

Claim 51 is directed to a data processing system comprising a data repository and a composite query operations manager residing in memory for managing execution of query operations in the data processing system. See *Application*, page 7, line 22 –

page 8, line 3. As claimed, the composite query operations manager is configured for receiving a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against the data repository. See *Application*, page 16, lines 10-27; FIG. 2. The composite query operations manager is also configured for retrieving selection logic defining a next query operation of the composite query operation to be executed. See *Application*, page 18, lines 3–15; page 23, line 23 – page 24, line 17; FIG. 3-4. The composite query operations manager is also configured for retrieving a plurality of failure conditions for determining when a failure of the composite query operation occurs. See *Application*, page 18, line 16 – page 20, line 4; page 24, lines 9-17; FIG. 3-4. The composite query operations manager is also configured for managing execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions. See *Application*, page 24, lines 13-17; page 26, line 23 – page 27, line 17; FIG. 4, 5B. The composite query operations manager is also configured for, upon determining the composite query operation has completed, storing and returning a result of the composite query operation. See *Application*, page 17, lines 1-7.

Grounds of Rejection to be Reviewed on Appeal

1. Rejection of claims 1-17, 26-41, 50-51 under 35 U.S.C. § 102(a) as being anticipated by *Bert Scalzo* "Oracle DBA Guide to Data Warehousing and Star Schema" (hereinafter *Scalzo*).

ARGUMENTS

1. Claims 1-17, 26-41, 50-51 are not anticipated by *Scalzo* under 35 U.S.C. § 102(a).

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990).

Regarding claims 1-8, 26-32, and 50:

Applicants submit that *Scalzo* does not teach a "method of managing execution of query operations in a data processing system" that includes "updating [an] operation status based on a result of [a] subsequent query operation," as recited by claim 1. As claimed, an initial "operation status" is determined for an "initial query operation." Based on the operation status, one of the plurality of subsequent query operations is selected and executed. After executing the subsequent query operation, method of claim 1 includes "updating the operation status based on a result of the subsequent query operation." Claims 26 and 50 recite similar operations. Further, claim 1 also recites updating the operation status based on a result of the subsequent query operation and managing execution of any remaining subsequent query operations based on the updated operation status.

The Examiner suggests that *Scalzo*, p. 7, discloses the limitation by recited by claim 1 of "updating the operation status based on a result of the subsequent query operation." At page 7, *Scalzo* provides source code for a C program used to illustrate an example of how the "Pro-C language supports "Dynamic SQL." The specific portions

cited by the Examiner include a while loop conditioned on a successful call to "fgets." In the soruce code cited by the Examiner, the "fgets" call is used to sequentially access database records, and the while loop processes each successive database record from within the body of the while loop. The Examiner suggests that this process discloses the claimed step of "updating the operation status based on a result of the subsequent query operation." Specifically, the Examiner suggests:

> Scalzo teaches in form of source codes that after each iteration of executing, the operation status is checked (by examining the result of "fgets") to determined whether to continue executing the while loop. The "fgets" command therefore serves a function to update the operation status.

Advisory Action, p. 2. However, the "fgets" command does not update the operation status of the subsequent query operation, as is required by claim 1. In fact, the "fgets" command does not update any sort of status at all and certainly not an "operation status" based on "a result of the subsequent query operation." Instead, the "while loop" uses the "fgets" command to simply read in the next record from an input file (*Scalzo,* page 7) until the last record is read. The condition included within the while loop is whether the well known "fgets" call was successfully used to read another record from the file. No "query operation" is perfomed by the "fgets" call, subsequent otherwise. Thus, the Applicants submit that *Scalzo* does not teach "updating [an] operation status based on a result of [a] subsequent query operation," as recited in the present claims.

Therefore, for all the foregoing reasons, claims 1, 26, and 50 are believed to be allowable, and allowance of the claims is respectfully requested. Furthermore, claims 2-8, and 27-32 depend upon claims 1 and 26, respectively. Applicants submit that these dependent claims are allowable and respectfully request allowance of the same.

Regarding claims 9-17, 33-41, and 51:

Applicants submit that *Scalzo* does not teach "managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of

failure conditions," as recited in claims 9, 33, and 51. Regarding this element, the Examiner suggests:

> Scalzo clearly teaches at pages 7-8 the step of executing an initial query operation (i.e. "update_command"), and based on the plurality of failure conditions (i.e, "if(sqlca.sqlcode == 1403)" and if (sqlca.sqlcode !=0)), performing the subsequent operation. Further, Scalso [sic] teaches the UPDATE and INSERT operations are performed within a "WHILE" loop so that multiple operations are performed while the conditions in the WHILE loop are true. Scalso [sic] therefore teaches at least an initial query and a plurality of subsequent query operations being executed based on the basis of selection logic and the plurality of failure conditions as claimed. The "fgets" command therefore serves as a function to update the operation status.

Advisory Action, p. 2. Here, the Examiner appears to suggest that executing a "plurality of subsequent query operations" is disclosed by passing through the while loop shown in *Scalzo* pages 7-8. However, Applicants respectfully submit that Scalzo in no way discloses that each pass through the while loop is performed "on the basis of the selection logic and the plurality of failure conditions," as recited in the present claims. In fact, the cited code does not disclose a "plurality of failure conditions" at all. Instead, the while loop is used to step through a set of database records using the "fgets" command. The Examiner is correct in suggesting that, for each record, an "UPDATE" operation is attempted, and if this operation fails, an "INSERT" operation is performed. Nothing in this "first try A, then try B" structure does the code cited by the Examiner carry out the claimed step of and "providing a plurality of failure conditions for determining when a failure of the composite query operation occurs" and "managing ... execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions." Instead, as demonstrated above, the "fgets" call included in the while loop condition does not update an operation status used to select subsequent query operations. Instead, the "while loop" uses the "fgets" command to simply read in the next record from an input file (*Scalzo,* page 7) until the last record is read. No "operation status" is updated, and therefore, no subsequent query operations can be "managed" based upon previous failure conditions, in the manner claimed. For example, claim 9 recites "managing, using a composite

query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions." This allows remaining query operations to be managed by the Composite Query Operations Manager (Fig. 3), based upon a plurality of failure conditions.

Applicants submit, therefore, that *Scalzo* does not teach "managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions," as recited in claims 9, 33, and 51. Accordingly, claims 9, 33, and 51 are believed to be allowable, and allowance of the claims is respectfully requested. Furthermore, claims 10-17, and 34-41 depend upon claims 9 and 33, respectively. Accordingly, for all of the reasons given above, Applicants submit that these dependent claims are allowable and respectfully request allowance of the same.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

CONCLUSION

The Examiner errs in finding that claims 1-17, 26-41, 50-51 are anticipated by *Scalzo* under 35 U.S.C. § 102(a).

Withdrawal of the rejections and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd., Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1. (Previously Presented) A method of managing execution of query operations in a data processing system, comprising:

issuing, by a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system;

executing the initial query operation;

determining an operation status of the initial query operation;

selecting one of the plurality of subsequent query operations based on the operation status;

performing the selected subsequent query operation;

updating the operation status based on a result of the subsequent query operation;

managing execution of any remaining subsequent query operations on the basis of the updated operation status; and

upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity.

2. (Original) The method of claim 1, wherein the determining and managing are performed by a composite query operations manager.

3. (Original) The method of claim 1, wherein the requesting entity is an application and wherein the determining and managing are performed by a composite query operations manager.

4. (Original) The method of claim 1, wherein the initial and the subsequent query operation are SQL statements.

5. (Original) The method of claim 1, wherein determining an operation status of the initial query operation comprises determining a number of items affected by the initial query operation.

6. (Original) The method of claim 1, wherein determining an operation status of the initial query operation comprises determining whether the initial query operation completed successfully.

7. (Original) The method of claim 1, wherein determining an operation status of the initial query operation comprises determining, on the basis of a return code received upon completion of the initial query operation, whether the initial query operation completed successfully.

8. (Original) The method of claim 1, wherein managing execution of the subsequent query operation comprises:

executing the subsequent query operation only if the initial query operation did not complete successfully.

9. (Previously Presented) A method of managing execution of query operations in a data processing system, comprising:

issuing, by a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system;

providing selection logic defining a next query operation of the composite query operation to be executed;

providing a plurality of failure conditions for determining when a failure of the composite query operation occurs;

managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions; and

upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity.

10. (Original) The method of claim 9, wherein at least one failure condition of the plurality of failure conditions indicates the initial query operation and an operation status of the initial query operation which indicates a failure of the composite query operation.

11. (Original) The method of claim 10, wherein each other failure condition of the plurality of failure conditions indicates a series of the initial query operation and at least one of the plurality of subsequent query operations, and an operation status of the at least one of the plurality of subsequent query operations which indicates a failure of the composite query operation.

12. (Original) The method of claim 9, wherein managing execution of the initial query operation and the plurality of subsequent query operations comprises:

a) executing the initial query operation;

b) determining an operation status of the initial query operation;

c) determining, on the basis of the operation status and the plurality of failure conditions, whether failure of the composite query operation occurred; and

d) if no failure of the composite query operation occurred:

i) determining the next operation to be executed from the plurality of subsequent query operations using the selection logic;

ii) executing the next query operation;

iii) determining an operation status of the next query operation;

iv) determining, on the basis of the operation status and the plurality of failure conditions, whether failure of the composite query operation occurred; and

v) repeating step d) for at least one other of the plurality of subsequent query operations.

13. (Original) The method of claim 12, wherein step c) and iv) comprise, if failure occurred:

completing execution of the composite query operation; and

returning a failure code as completion status of the composite query operation indicating a failure condition from the plurality of failure conditions.

14. (Original) The method of claim 12, wherein step b) and step iii) comprise determining whether the respective query operation completed successfully.

15. (Original) The method of claim 12, wherein step b) and iii) comprise determining, on the basis of a return code received upon completion of the respective query operation, whether the respective query operation completed successfully.

16. (Original) The method of claim 12, wherein step b) and step iii) comprise determining whether the respective query operation completed successfully and, if the respective query operation completed successfully:

completing execution of the composite query operation.

17. (Original) The method of claim 9, wherein the initial and each subsequent query operation is an SQL statement.

18 – 25 (Cancelled)

26. (Previously Presented) A computer readable storage medium containing a program which, when executed, performs a process of managing execution of query operations in a data processing system, the process comprising:

receiving, from a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system;

executing the initial query operation;

determining an operation status of the initial query operation;

selecting one of the plurality of subsequent query operations based on the operation status;

performing the selected subsequent query operation;

updating the operation status based on a result of the subsequent query operation;

managing execution of any remaining subsequent query operations on the basis of the updated operation status; and

upon determining the composite query operation has completed, returning a result of the composite query operation to the requesting entity.

27. (Previously Presented) The computer readable storage medium of claim 26, wherein the determining and managing are performed by a composite query operations manager.

28. (Previously Presented) The computer readable storage medium of claim 26, wherein the initial and the subsequent query operation are SQL statements.

29. (Previously Presented) The computer readable storage medium of claim 26, wherein determining an operation status of the initial query operation comprises determining a number of items affected by the initial query operation.

30. (Previously Presented) The computer readable storage medium of claim 26, wherein determining an operation status of the initial query operation comprises determining whether the initial query operation completed successfully.

31. (Previously Presented) The computer readable storage medium of claim 26, wherein determining an operation status of the initial query operation comprises determining, on the basis of a return code received upon completion of the initial query operation, whether the initial query operation completed successfully.

32. (Previously Presented) The computer readable storage medium of claim 26, wherein managing execution of the subsequent query operation comprises:

executing the subsequent query operation only if the initial query operation did not complete successfully.

33. (Previously Presented) A computer readable storage medium containing a program which, when executed, performs a process of managing execution of query operations in a data processing system, the process comprising:

receiving, from a requesting entity, a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against a data repository of the data processing system;

retrieving selection logic defining a next query operation of the composite query operation to be executed;

retrieving a plurality of failure conditions for determining when a failure of the composite query operation occurs;

managing, using a composite query operations manager, execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions; and

upon determining the composite query operation has completed, storing and returning a result of the composite query operation to the requesting entity.

34. (Previously Presented) The computer readable storage medium of claim 33, wherein at least one failure condition of the plurality of failure conditions indicates the initial query operation and an operation status of the initial query operation which indicates a failure of the composite query operation.

35. (Previously Presented) The computer readable storage medium of claim 34, wherein each other failure condition of the plurality of failure conditions indicates a series of the initial query operation and at least one of the plurality of subsequent query

operations, and an operation status of the at least one of the plurality of subsequent query operations which indicates a failure of the composite query operation.

36. (Previously Presented) The computer readable storage medium of claim 33, wherein managing execution of the initial query operation and the plurality of subsequent query operations comprises:

a) executing the initial query operation;

b) determining an operation status of the initial query operation;

c) determining, on the basis of the operation status and the plurality of failure conditions, whether failure of the composite query operation occurred; and

d) if no failure of the composite query operation occurred:

i) determining the next operation to be executed from the plurality of subsequent query operations using the selection logic;

ii) executing the next query operation;

iii) determining an operation status of the next query operation;

iv) determining, on the basis of the operation status and the plurality of failure conditions, whether failure of the composite query operation occurred; and

v) repeating step d) for at least one other of the plurality of subsequent query operations.

37. (Previously Presented) The computer readable storage medium of claim 36, wherein step c) and iv) comprise, if failure occurred:

completing execution of the composite query operation; and

returning a failure code as completion status of the composite query operation indicating a failure condition from the plurality of failure conditions.

38. (Previously Presented) The computer readable storage medium of claim 36, wherein step b) and step iii) comprise determining whether the respective query operation completed successfully.

39. (Previously Presented) The computer readable storage medium of claim 36, wherein step b) and iii) comprise determining, on the basis of a return code received upon completion of the respective query operation, whether the respective query operation completed successfully.

40. (Previously Presented) The computer readable storage medium of claim 36, wherein step b) and step iii) comprise determining whether the respective query operation completed successfully and, if the respective query operation completed successfully:

completing execution of the composite query operation.

41. (Previously Presented) The computer readable storage medium of claim 33, wherein the initial and each subsequent query operation is an SQL statement.

42- 49. (Cancelled)

50. (Previously Presented) A data processing system comprising:

a data repository; and

a composite query operations manager residing in memory for managing execution of query operations in the data processing system, the composite query operations manager being configured for:

receiving a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against the data repository;

executing the initial query operation;

determining an operation status of the initial query operation;

selecting one of the plurality of subsequent query operations based on the operation status;

performing the selected subsequent query operation;

updating the operation status based on a result of the subsequent query operation;

managing execution of any remaining subsequent query operations on the basis of the updated operation status; and

upon determining the composite query operation has completed, storing and returning a result of the composite query operation to the requesting entity.

51. (Previously Presented) A data processing system comprising:

a data repository; and

a composite query operations manager residing in memory for managing execution of query operations in the data processing system, the composite query operations manager being configured for:

receiving a request to perform a composite query operation defined by at least an initial query operation and a plurality of subsequent query operations to be executed against the data repository;

retrieving selection logic defining a next query operation of the composite query operation to be executed;

retrieving a plurality of failure conditions for determining when a failure of the composite query operation occurs;

managing execution of the initial query operation and the plurality of subsequent query operations on the basis of the selection logic and the plurality of failure conditions; and

upon determining the composite query operation has completed, storing and returning a result of the composite query operation.

52 – 54. (Cancelled)

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.